International Journal of Engineering and Technical Research (IJETR)
ISSN: 2321-0869 (O) 2454-4698 (P) Volume-7, Issue-10, October 2017

A Novel Approach for Multiple Vertical Fragmentations of Datasets Using Affinity Matrix in Distributed Environment: A Survey

Sunil Kumar Verma, Dr. Neelendra Badal

Abstract— **Designing an efficient Distributed Database System (DDBS) is considered as one of the most challenging problems because of multiple interdependent factors which are affecting its performance. Allocation and fragmentation are two processes which their efficiency and correctness influence the performance of DDBS. Therefore, efficient data fragmentation and allocation of fragments across the network sites are considered as an important research area in distributed database design. In this paper presents an approach which simultaneously fragments data vertically and allocates the fragments to appropriate sites across the network. Bond Energy Algorithm (BEA) is applied with a better affinity measure that improves the generated clusters of attributes.**

Index Terms— **BEA, Fragmentation, Distributed Database System, Slop Based Partitioning Algorithm.**

I. INTRODUCTION

In distributed computing environments, each unit of data (item) which is accessed at the station, (site) is not usually a relationship but part of the relationship. Therefore, to optimize the performance of the query, the relations of global schema are fragmented into items.

The primary concern of distributed database systems is to design the fragmentation and allocation of the underlying database. The distribution design involves making decisions on the fragmentation and placement of data across the sites of a computer network. The first phase of the distribution design in a top-down approach is the fragmentation phase, which is the process of clustering into fragments the information accessed simultaneously by applications. The fragmentation phase is then followed by the allocation phase, which handles the physical storage of the generated fragments among the nodes of a computer network, and the replication of fragments.

A Distributed database [1] is a database that is under the control of a central database management system (DBMS) in which storage devices are not all attached to a common CPU. It may be stored in multiple computer located in the same physical location, or may be dispersed over a network of interconnected computers. There are multiple sites (computers) in a distributed database so if one site fails then system will not be useless, because other sites can do their job because same copy of data is installed on every location

Sunil Kumar Verma, Asst. Professor (Sr. Scale), Department Of Computer Science & Engineering, Feroze Gandhi Institute of Engineering& Technology Raebareli UP

Dr. Neelendra Badal, Associate Professor, Department Of Computer Science & Engineering, Kamala Nehru Institute of Engineering & Technology Sultanpur UP

II. RELATED WORK

Most of the vertical splitting algorithms have started from constructing an attribute affinity matrix from the attribute usage matrix: the Attribute affinity matrix is an m x m matrix for the m-attribute problem whose (i, j) element equals the “between attributes” affinity which is the total number of accesses of transactions referencing both attributes i and j. An iterative binary partitioning method has been used in [8] and [5] based on first clustering the attributes and then applying empirical objective functions or mathematical cost functions to perform the fragmentation. The concept of using fragmentation of data as a means of improving the performance of a database management system has often appeared in the literature on file design and optimization. Attribute partitioning and attribute clustering have been studied earlier by [4], [3], [6], [8], [9] has discussed the implementation of a self-reorganizing database management system that carries out attribute clustering. They also show that in a database management system where storage cost is low compared to the cost of accessing the sub files, it is beneficial to cluster the attributes, since the increase in storage cost will be more than offset by the saving in access cost. Hoffer [6] developed a non-linear, zero-one program, which minimizes a linear combination of storage, retrieval and update costs, with capacity constraints for each file.

Navathe et al [8] used a two-step approach for vertical partitioning. In the first step, they used the given input parameters in the form of an attribute usage matrix to construct the attribute affinity matrix on which clustering is performed. After clustering, an empirical objective function is used to perform iterative binary partitioning. In the second step, estimated cost factors reflecting the physical environment of fragment storage are considered for further refinement of the partitioning scheme. Cornell and Yu [5] proposed an algorithm, as an extension of Navathe et al [8] approach, which decreases the number of disk accesses to obtain an optimal binary partitioning. This algorithm uses specific physical factors such as number of attributes, their length and selectivity, cardinality of the relation etc.

Navathe and Ra have developed a new algorithm based on a graphical technique [7]. This algorithm starts from the attribute affinity matrix by considering it as a complete graph called the “affinity graph” in which an edge value represents the affinity 1-4244-1364-8/07/$25.00 ©2007 IEEE between the two attributes, and then forms a linearly connected spanning tree. The algorithm generates all meaningful fragments in single iteration by considering a cycle as a fragment. A linearly connected tree has only two ends. By a “linearly connected tree” we imply a tree that is constructed by including one edge at a time such that only edges at the

"first" and the "last" node of the tree would be considered for inclusion. We then form "affinity cycles" in this spanning tree by including the edges of high affinity value around the nodes and "growing" these cycles as large as possible. After the cycles are formed, partitions are easily generated by cutting the cycles apart along "cut-edges". In this paper we will use an algorithm to cluster the database i.e. Bond Energy Algorithm (BEA). And use these cluster affinity as input to find final fragments using PARTITION algorithm. Then using prototypes we reach to the goal of reducing response time of query using fragmentation and show the mathematical result for proof.

III. DDBMS

The design of a distributed database system involves making decisions on the architecture of DDBMS . Two major strategies proposed by Ceri and pelagatti for designing distributed databases are : top-down approach and bottom-up approach. In the case of tightly integrated distributed database design proceed stop-down form requirements analysis and logical design of the global database to physical design of each local database. In the case of distributed multi database systems, the design process is bottom-up and involves the integration of exiting databases. But real applications are rarely simple enough to fit nicely in either of these approaches. The two approaches may need to be applied together to complement each other.

A. Top-down Approach

In the top-down approach, the process starts with a requirement Analysis that defines the environment to the system and elicits both the data and processing needs of all potential database users [8]. The requirements analysis also specifies where the final system is expected to stand with respect to the objectives of the DDBMS. The objective is defined with respect to performance, reliability and availability, economics, and expandability (flexibility).The requirement documents are the inputs to two parallel activities: view design and Conceptual design. The outputs of view design are the user, and the output of conceptual design is entity types and relationship types which are used to construct an externals schema.

B. Bottom-Up Approach

The top-down design is suitable for the systems which are developed from scratch. But when the distributed data base is developed as the aggregation of exiting databases, it is not easy to follow the top-down approach. The bottom-up approach, which starts with individual local conceptual schemata, is more suitable for this environment explained that the bottom-up approach is based on the integration of existing schemata into a single, global schema. Integration is the process of the merging of common data definitions and the resolution of conflicts among different representation that are given to the same data. The global conceptual schema is the product of the process concluded that there are three requirements for bottom-up design.

1. The selection of a common database model for describing the global schema of the database.
2. The transaction of each local schema into the common data model.
3. The integration of the local schema into a common global schema.

IV. BACKGROUND OF SPLITTING ALGORITHM

Today, mostly centralized databases are used to store and manage data [11]. They carry the advantages of high degree of security, concurrency, backup and recovery control. However, they also have the disadvantages of high communication costs (when the client is far away and communication is very frequent in between the client & server), unavailability in case of system failure and a single source bottleneck [3].

Research conducted in 1991 for distributed databases predicted a huge shift from traditional databases to distributed databases in the coming arena, due to organizational needs to manage huge amounts of data [11]

The telecommunication sector also wants to embrace this technology of data distribution. But before distribution of data, fragmentation is a very important and critical task that needs to be done.

Most of the telecom industries are using centralized technique in storage of their database. Centralized database has its disadvantage of high communication cost. Some data is unavailable due to problem in server. To resolve these issues we are moving from centralized database to distribution of the database.

V. DATABASE FRAGMENTATION

Multiple Vertical fragmentation schemes is being described. Further this chapter is going to describe the clustering method to make the cluster of attributes of a database table. In this chapter Bond Energy Algorithm (BEA) is going to be used for clustering of attributes. Further a new algorithm is being described for fragmentation of clusters of attributes of database table. Vertical partitioning is used during design of a database to enhance the performance of query execution. In order to obtain improved performance, fragments must closely match the requirements of the query workload. Vertical partitioning involves splitting the relations along the columns into partitions all having equal number of rows. Each partition now acts as a separate relation but we preserve the row ordering in all the partitions as it was in the original relation. It should be noted that each partition may contain more than one column. However, when columns in multiple partitions are accessed instead of join we just need to do pasting of columns.

The advantages of multiple vertical partitioning are as follows: If query involves only few columns then we avoid unnecessary fetching of other columns. This saves the I/O bandwidth and avoids unnecessary processing. Moreover data in a column belongs to the same domain e.g., values in salary column will be numeric within some range. This similarity in data can be well exploited by compression algorithms and better compression ratios can be achieved.

The data under consideration is of the order of terabytes. Since the data is large enough to be handled by traditional relational database system we resort to file system for storage. Each partition as described above is stored as a separate file. Since the row ordering is maintained we do not store the row identifiers in the file. The meta data stores information regarding the partitions of a given relation such as number of partitions, columns in each.

Vertical fragmentation: Subsets of attributes (that is, columns) form the fragments. Rows of the fragments that correspond to each other have to be linked by a tuple identifier. A vertical fragmentation corresponds to projection operations on the table. [2]

Horizontal fragmentation: Subsets of tuples (that is, rows) form the fragments. A horizontal fragmentation can be expressed by a selection condition on the table.

Derived fragmentation: A given horizontal fragmentation on a primary table (the primary fragmentation) induces a horizontal fragmentation of another table based on the semi join with the primary table.
In this case, the primary and derived fragments with matching values for the join attributes can be stored on the same server; this improves efficiency of a join on the primary and the derived fragments.

The following three properties are considered the important correctness properties of a fragmentation:

Completeness: No data should be lost during fragmentation. For vertical fragmentation, each column can be found in some fragment; in horizontal fragmentation each row can be found in a fragment.

Reconstructability: Data from the fragments can be recombined to result in the original data set. For vertical fragmentation, the join operator is used on the tuple identifier to link the columns from the fragments; in horizontal fragmentation, the union operator is used on the rows coming from the fragments.

Non-redundancy: To avoid duplicate storage of data, data should be uniquely assigned to one fragment. In vertical fragmentation, each column is contained in only one fragment (except for the tuple identifier that links the fragments); in horizontal fragmentation, each row is contained in only one fragment.

In this paper we will compute semantically-guided multiple vertical fragmentations of a primary table. Each of these fragmentations will be based on clustering an attribute for which values should be relaxed to allow for flexible query answering. In contrast to the conventional applications of fragmentation, the clustering-based fragmentations will support flexible query answering in an efficient manner.
Fragmentation and allocation are usually performed separately while these two steps of Distributed DBMS design are closely related to each other. The reason for separating the distribution design into two steps is to better deal with the complexity of the problem [12].
Here we present a method for VF, which applies BEA hierarchically with a modified similarity measure and simultaneously allocates the fragments to the most appropriate site.

VI. METHODS

Bond Energy Algorithm (BEA) has been used for clustering of entities. BEA finds an ordering of entities (in our case attributes) such that the global affinity measure is maximized.

$$AM = \sum_i \sum_j (affinity\ of\ A_i\ and\ A_j\ with\ their\ neighbors)$$

- The bond energy algorithm (BEA) was developed and has been used in the database design area to determine how to group data and how to physically place data on a disk.
- It can be used to cluster attributes based on usage and then perform logical or physical design accordingly. With BEA, the affinity (bond) between database attributes is based on common usage.
- This bond is used by the clustering algorithm as a similarity measure. The actual measure counts the number of times the two attributes are used together in a given time. To find this, all common queries must be identified.
- The idea is that attributes that are used together form a cluster and should be stored together. In a distributed database, each resulting cluster is called a vertical fragment and may be stored at different sites from other fragments.

Allocation and fragmentation are interdependent problems where solving them simultaneously is difficult but results in better performance of applications. To the best of our knowledge, BEA is not applied to simultaneous fragmentation and allocation.
Since in vertical partitioning attributes which are usually accessed together are placed in one fragment, defining a precise measure of togetherness is critical. BEA uses affinity of attributes to create clusters of attributes, which are the most similar.
It starts with Attribute Usage (AU) and Query Access (QA) matrices generates Attribute Affinity matrix (AFF) and finally creates Clustered Affinity matrix (CA) by positioning and repositioning columns and rows of attributes. The affinity measure is too simple.
The basic steps of this clustering algorithm are:

i. Create an attribute affinity matrix in which each entry indicates the affinity between the two associate attributes. The entries in the similarity matrix are based on the frequency of common usage of attribute pairs.
ii. The BEA then converts this similarity matrix to a BOND matrix in which the entries represent a type of nearest neighbor bonding based on probability of co-access. The BEA algorithm rearranges rows or columns so that similar attributes appear close together in the matrix.
iii. Finally, the designer draws boxes around regions in the matrix with high similarity.

The resulting matrix, modified from, is illustrated in Figure 1. The two shaded boxes represent the attributes that have been grouped together into two clusters.

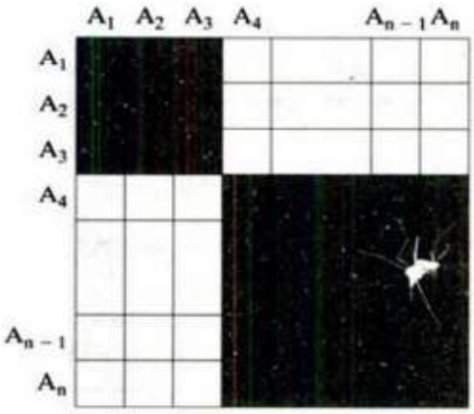

Figure 1: Clustered Affinity Matrix for BEA

Two attributes Ai and Aj have a high affinity if they are frequently used together in database applications. At the heart of the BEA algorithm is the global affinity measure. Suppose that a database schema consists of n attributes {A1, A2, , An}. The global affinity measure, AM, is defined as

$$AM = \sum_{i=1}^{n} (bond(A_i, A_{i-1}) + bond(A_i, A_{i+1}))$$

VII. Conclusion

Distributed databases reduce cost of update and retrieval of information and increase performance and availability, but the design of DDBMS is more complicated than designing centralized database. One of the major challenges which greatly affect DDBS performance is fragmentation and allocation of fragments to sites. Fragmentation can logically be merged and done simultaneously. In this paper we proposed a method that merges vertical fragmentation. To achieve this goal we applied Bond Energy Algorithm with a modified affinity measure in a hierarchical process and simultaneously calculated the cost of data allocation for each site and assigned fragment to the appropriate site. The use of the hierarchical process resulted in clustering sets of more similar attributes and better data fragmentation. On the other hand, by performing simultaneous cost calculation we took interdependency of data fragmentation and allocation into account.

An extension to the work could cover optimizing the cost function for data allocation considering the retrieval and update frequency for each attribute and applying better approach to calculate weights for similarity measure.

References

[1] Ceri, S. and Pelagatti, G. Distributed DatabasesPrinciples and Systems. NY, McGraw Hill, 1984. .

[2] Ezeife, C. I. and Barker, K. Vertical Class Fragmentation in a Distributed Object Based Svstem. TR 94-03, Univ. of Manitoba DeRt. ofCbmputer Science, 1993.

[3] H.o ffer. 1. A.. and Severance. D. G. The Use of Cluster Analysis in Physical Database Design.In Proceedings of 1st VLDB Conference, Mass., 1975.

[4] Karlapalem, K. and Li, 8. Partitioning Schemes for Object Oriented Database. In 5th InternationalWorkshop on Research Issues on Data Engineering: Distributed Object Management, 1995.

[5] Karlapalem, K., Li, 8. and Vieweg,, S. Method Induced Partitioning Schemes in Object OrientedDatabases. In 16th intemational conference on Distributed Computing System, Hong Kong, 1996.

[6] Karlapalem, K., Navathe, S. B. and Morsi, M. M.A. Issues in Distribution design. of object-oriented databases, in Distributed Object Management, Morgan Kaufmann Publishers, 1994.

[7] Lee, S. and Lim, H., Extension of Vertical Technical Conference on Circuits/systems, Computers And Communications, Japan, 1997.

[8] Navathe, S. B., Ceri, S. Wiederhold, G. and Dou, J.Vertical partitioning algorithms for database design.in ACM TODS 9(4), 1984.

[9] Farhi Marir, Yahiya Najjar, Mahmoud Y. AlFaress, Hassan I. Abdalla, “An Enhanced Grouping Algorithm for Vertical Partitioning Problem in DDBs.

[10]Wiederhold, G., and Dou, J.,“Vertical Partitioning Algorithms for Database Design,” ACM Trans.on Database Systems, Vol. 9, No.4, Dec. 1984.

[11]Ashraf, Imran And Khokhar, A.S. 2010. Principles for Distributed Databases in Telecom Environment., Sweden.

[12]M.T. Ozsu, P. Valduriez, Principles of Distributed Database Systems, Alan Apt, New Jersey, 1999.

ABOUT THE AUTHOR

Sunil Kumar Verma Asst. Professor (Sr. Scale) Department Of Computer Science & Engineering, Feroze Gandhi Institute of Engineering & Technology Raebareli U.P. Sunil Kumar Verma is a Ph.D Scholar Department of Computer Science & Engineering at Himalayan University Itanagar Arunachal Pradesh(India. He has received Masters degree M.Tech (Software System) , Department Of Computer Science & Engineering from Birala Institute of Technology & Science (BITS) PILANI Rajasthan India and he has received his Bachelor of Technology degree from Bundelkhand Institute of Technology (BIET), Jhansi in Computer Science & Engineering Pradesh(India) ,

Dr. Neelendra Badal is an Associate Professor in the Department of Computer Science & Engineering at Kamla Nehru Institute of Technology (KNIT), Sultanpur (U.P), India. He received B.E. (1997) from Bundelkhand Institute of Technology (BIET), Jhansi in Computer Science & Engineering, M.E. (2001) in Communication, Control and Networking from Madhav Institute of Technology and Science (MITS), Gwalior and PhD (2009) in Computer Science & Engineering from Motilal Nehru National Institute of Technology (MNNIT), Allahabad. He is Chartered Engineer (CE) from Institution of Engineers (IE), India. He is a Life Member of IE, IETE, ISTE and CSI-India. He has published about 60 papers in International/National Journals, conferences and seminars. His research interests are Distributed System, Parallel Processing, GIS, Data Warehouse & Data mining, Software engineering and Networking